J. W. PRICE 949/261.8433
TAKAKAZU SHIOMI et al
NAKI-B082

S.N. 09/854,268

日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２０００年　５月１５日

出願番号
Application Number: 特願２０００－１４１４９３

出願人
Applicant (s): 松下電器産業株式会社

２００１年　１月１９日

特許庁長官
Commissioner,
Patent Office 及川耕造

出証番号　出証特２０００－３１１２８９７

【書類名】　特許願
【整理番号】　2022520224
【提出日】　平成12年 5月15日
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/46
【発明者】
　【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
　【氏名】　塩見　隆一
【発明者】
　【住所又は居所】　広島県東広島市鏡山３丁目１０番１８号　株式会社松下電器情報システム広島研究所内
　【氏名】　久保岡　祐子
【特許出願人】
　【識別番号】　000005821
　【氏名又は名称】　松下電器産業株式会社
【代理人】
　【識別番号】　100097445
　【弁理士】
　【氏名又は名称】　岩橋　文雄
【選任した代理人】
　【識別番号】　100103355
　【弁理士】
　【氏名又は名称】　坂口　智康
【選任した代理人】
　【識別番号】　100109667
　【弁理士】
　【氏名又は名称】　内藤　浩樹

【手数料の表示】

【予納台帳番号】　011305

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9809938

【書類名】　明細書

【発明の名称】　Ｊａｖａアプリケーション実行装置

【特許請求の範囲】

【請求項１】　複数のスレッドで構成されるタスク単位で資源回収を行うＯＳと、

Ｊａｖａアプリケーションを記憶するアプリケーション記憶手段と、

前記アプリケーション記憶手段が記憶するＪａｖａアプリケーションを実行するＪａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅと、

前記Ｊａｖａアプリケーションを前記ＯＳが生成するタスク及びスレッドと組にして管理し、終了するアプリケーションが使用している資源を前記ＯＳに回収する指示を出すアプリケーション管理手段と、

前記Ｊａｖａアプリケーション毎にスレッドを生成し、生成したスレッド上で前記アプリケーションのメソッドを呼び出すクラスライブラリを備えることを特徴とするＪａｖａアプリケーション実行装置。

【請求項２】　前記Ｊａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅは、タスク・スレッドを生成し、生成したスレッド上でＪａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅが使用する資源を確保することを特徴とする請求項１記載のＪａｖａアプリケーション実行装置。

【請求項３】　前記アプリケーション管理手段は、スレッドによってＪａｖａアプリケーションを特定し、

前記クラスライブラリは、前記Ｊａｖａアプリケーションに対して共通のスレッド生成し管理することを特徴とする請求項１記載のＪａｖａアプリケーション実行装置。

【請求項４】　前記アプリケーション管理手段は、Ｊａｖａアプリケーションの終了を前記クラスライブラリに通知し、

前記クラスライブラリは、アプリケーション終了時に保持しているＪａｖａアプリケーションに関するデータを削除することを特徴とする請求項３記載のＪａｖａアプリケーション実行装置。

【請求項５】　前記ＯＳは、前記クラスライブラリがＪａｖａアプリケーショ

ンを呼び出すタイミングを通知し、

前記クラスライブラリは、生成するスレッドに対応してキューを生成し、前記ＯＳからの通知によって、Ｊａｖａアプリケーションに通知する情報を前記キューに代入し、前記生成されたスレッドは前記キューに代入された情報を取り出し、Ｊａｖａアプリケーションを呼び出すことを特徴とする請求項１記載のＪａｖａアプリケーション実行装置。

【請求項６】　前記クラスライブラリは、Ｊａｖａアプリケーションからのリスナー登録時に前記Ｊａｖａアプリケーションに対応するスレッドを生成することを特徴とする請求項１記載のＪａｖａアプリケーション実行装置。

【請求項７】　前記クラスライブラリは、ＪａｖａアプリケーションがＣｏｍｐｏｎｅｎｔクラスのインスタンスを生成する時に前記Ｊａｖａアプリケーションに対応するスレッドを生成することを特徴とする請求項１記載のＪａｖａアプリケーション実行装置。

【請求項８】　前記クラスライブラリは、Ｊａｖａアプリケーションの起動時に前記Ｊａｖａアプリケーションに対応するスレッドを生成することを特徴とする請求項１記載のＪａｖａアプリケーション実行装置。

【請求項９】　コンピュータ読み取り可能な記録媒体であって、

複数のスレッドで構成されるタスク単位で資源回収を行うＯＳと、

Ｊａｖａアプリケーションを記憶するアプリケーション記憶手段と、

前記アプリケーション記憶手段が記憶するＪａｖａアプリケーションを実行するＪａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅと、

前記Ｊａｖａアプリケーションを前記ＯＳが生成するタスク及びスレッドと組にして管理し、終了するアプリケーションが使用している資源を前記ＯＳに回収する指示を出すアプリケーション管理手段と、

前記Ｊａｖａアプリケーション毎にスレッドを生成し、生成したスレッド上で前記アプリケーションのメソッドを呼び出すクラスライブラリの各機能を発揮するプログラムを記録したコンピュータ読み取り可能な記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、Ｊａｖａアプリケーションが終了時に解放しなかった資源の回収機能を備えるＪａｖａアプリケーション実行装置に関する。

【０００２】

【従来の技術】

従来のアプリケーション実行装置における資源回収方法は、特開平８－１２３７００「資源管理方法」に記述されている。アプリケーションはタスクに割り当てられ、タスクは複数のスレッドを有しアプリケーションを実行する。アプリケーションが使用する資源は、スレッドに関するものはスレッド単位で管理され、タスクに関する物はタスク単位で管理され、システム全体に関する物は、システム全体で管理される。図１１は、これら管理状態を図示したものである。空間１１０１はシステム全体を表し、システム全体に関する資源は空間資源１１０４で管理される。システム中には複数のタスク１１０２が存在し、タスクに関する資源は、タスク資源１１０５で管理される。タスク１１０２は複数のスレッド１１０３を有し、スレッドに関する資源はスレッド資源１１０６で管理される。管理されている資源は、スレッド終了時にはスレッドに関する資源だけが回収され、タスク終了時にはタスクに関する資源とその中に含まれるスレッドに関する資源が回収さる。

【０００３】

一方、Ｊａｖａアプリケーションを実行するＪａｖａミドルウエアは、様々なＯＳ上に実現されている。現在、Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ社から提供されている、Ｊａｖａミドルウエアは、Ｊａｖａアプリケーションの起動と同時に起動され、Ｊａｖａアプリケーションの終了と共に終了する。Ｊａｖａミドルウエア自体でＪａｖａアプリケーションが使用した資源回収は行わず、Ｊａｖａミドルウエアが終了することで、ＯＳによる資源回収が行われている。

【０００４】

【発明が解決しようとする課題】

一般にＪａｖａミドルウエアの起動処理は、最初に多くのクラスをメモリーに読み込むため時間がかかる。よって、Ｊａｖａアプリケーションのみを実行する

システムで、アプリケーションの起動、終了毎にミドルウエアの終了・再起動を行うと切り替え時間が長くなる。ミドルウエアを再起動せずに次のアプリケーションを実行すると資源回収が不十分なため、資源不足でアプリケーションが動作しなくなるという問題点がある。また、Ｊａｖａミドルウエアは、複数のＪａｖａアプリケーションに対して共通のスレッドを用意し、そのスレッドでアプリケーションから登録されたリスナー呼び出しやアプリケーションの描画メソッド呼び出しを行う。この共通スレッドで実行されるＪａｖａアプリケーションのコード中でシステム資源を確保し使用する場合が有るため、ＯＳレベルでのスレッド単位での資源管理が不可能という問題もある。

【０００５】

本発明は、Ｊａｖａミドルウエアにおいてアプリケーション毎に専用スレッドを生成しリスナー呼び出しやアプリケーションの描画メソッド呼び出しを行うことにより、ＯＳが備えるタスク単位の資源回収機能を利用した資源回収機能を実現し、再起動なしで長時間安定動作するＪａｖａミドルウエアを搭載したＪａｖａアプリケーション実行装置を提供する。

【０００６】

【課題を解決するための手段】

上記課題を解決するため本発明は、複数のスレッドで構成されるタスク単位で資源回収を行うＯＳと、Ｊａｖａアプリケーションを記憶するアプリケーション記憶手段と、前記アプリケーション記憶手段が記憶するＪａｖａアプリケーションを実行するＪａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅと、前記Ｊａｖａアプリケーションを前記ＯＳが生成するタスク及びスレッドと組にして管理し、終了するアプリケーションが使用している資源を前記ＯＳに回収する指示を出すアプリケーション管理手段と、前記Ｊａｖａアプリケーション毎にスレッドを生成し、生成したスレッド上で前記アプリケーションのメソッドを呼び出すクラスライブラリを備えることとしている。

【０００７】

【発明の実施の形態】

以下、本発明に係るアプリケーション実行装置の具体的な実施の形態を、図面

を参照しながら説明する。

【０００８】

（実施の形態１）

図１は、本発明に係るＪａｖａアプリケーション実行装置の形態を表したものであり、アプリケーション入力部１０１、アプリケーション記憶部１０２、Ｊａｖａミドルウエア１０３、ＯＳ１０４、ハードウエア１０５、制御部１０６で構成される。

【０００９】

アプリケーション入力部１０１は、実行すべきＪａｖａアプリケーションを受け付ける。具体的にはフロッピーディスクやＣＤ等のメディアを通してＪａｖａアプリケーションが供給される場合、本構成要素はフロッピーディスクドライブ、ＣＤドライブ等で構成される。またネットワークや放送局らからの電波をを通してＪａｖａアプリケーションが配信される場合は、ネットワークインターフェースボードや放送受信機器で構成される。内蔵されるＪａｖａアプリケーションのみを実行する装置においては、アプリケーション記憶部１０２にＪａｖａアプリケーションを記憶しておくだけで良いので、本要素は不要となる。

【００１０】

アプリケーション記憶部１０２は、実行すべきＪａｖａアプリケーションを記憶する。具体的には、ＲＡＭ、ＲＯＭ、ハードディスクなどの内蔵機器や、ＣＤドライブ、フロッピーディスクや、フロッピーディスクなどのリムーバブルメディア等で構成される。

【００１１】

Ｊａｖａミドルウエア１０３は、アプリケーション記憶部１０２が記憶するＪａｖａアプリケーションを実行する。内部には、ＶＭ部１０３ａ、アプリケーション管理部１０３ｂ、クラスライブラリ格納部１０３ｃを構成要素として持つ。

【００１２】

ＶＭ部１０３ａはアプリケーション記憶部１０２が記憶するＪａｖａアプリケーションを実行する。Ｊａｖａアプリケーションを構成するバイトコードを逐次解析し実行する。Ｊａｖａアプリケーションは、システムが備えているデバイス

を制御するため、デバイスを制御するためのクラスライブラリを呼び出す。ＶＭ部１０３ａは、Ｊａｖａアプリケーションがクラスライブラリを呼び出すとき、クラスライブラリ格納部１０３ｃが格納するクラスライブラリから必要なものを呼び出し実行する。また、Ｊａｖａアプリケーションはスレッドを生成したりオブジェクトのインスタンスを生成するためメモリーを使用する。この場合、ＶＭ部１０３ａは後述するＯＳ１０４のカーネル１０４ａを呼び出し、スレッドを生成しメモリーを確保する。

【００１３】

アプリケーション管理部１０３ｂは、ＶＭ部１０３ａが実行するＪａｖａアプリケーションを管理し、以下の３つの機能を有する。

【００１４】

Ｊａｖａアプリケーションに対応するタスクとタスク中の最初のスレッドＯＳ１０４のカーネル１０４ａに依頼して作成し、ＶＭ部１０３ａに依頼して、そのスレッド上でＪａｖａアプリケーションを実行する。この際、アプリケーションとタスクを対応付けて保持する。また、動作しているＪａｖａアプリケーションから派生して生成される全てのスレッドを管理する。

【００１５】

図２は、アプリケーション管理部１０３ｂがアプリケーションとタスクとスレッドを対応付けて管理しているテーブルの例である。図２（１）では２つのアプリケーションが存在し、アプリケーション管理部１０３ｂは、２つのアプリケーションにアプリケーションＩＤ「１」，「２」を割り当てている。各アプリケーションにはＯＳによって用意されたタスクのタスクＩＤ「２０１」「２０２」が保持されている。アプリケーションＩＤ「１」のアプリケーションは２つのスレッドで実行されており、それらのスレッドのスレッドＩＤ「１」「２」を保持している。アプリケーションＩＤ「２」のアプリケーションは３つのスレッドで実行されており、それらのスレッドのスレッドＩＤ「４」「５」「６」を保持している。ここで、アプリケーションＩＤ「１」のアプリケーションがスレッドを生成すると、追加されたスレッドのスレッドＩＤがテーブルに追加される。図２（２）は、生成されたスレッドのスレッドＩＤ「７」が追加された状態を表す。ま

た、図２（２）の状態で、スレッドＩＤ「５」のスレッドがアプリケーションにより削除されると、図２（３）のようテーブルからスレッドＩＤ「５」が削除される。

【００１６】

アプリケーションが終了する際、カーネル１０４ａに終了するアプリケーションを実行しているタスクを通知し、ＯＳ１０４による資源回収を依頼する。またクラスライブラリ格納部１０３ｃが格納するクラスライブラリにアプリケーションが終了することを通知する。その後、アプリケーションを実行しているタスク及びその中に含まれる全てのスレッドを終了し、ＶＭ部１０３ａによる実行を停止する。

【００１７】

例えば、アプリケーション管理部１０３ｂが図２（３）の状態で、アプリケーションＩＤ「2」のアプリケーションを終了させる場合、まずカーネル１０４ａに対応するタスクＩＤ「２０２」を通知し、アプリケーションに関する資源回収を依頼する。資源回収が終了したのち、アプリケーションに対応するタスクＩＤ「２０２」ｎｏタスク及びその中に含まれるスレッドＩＤ「４」，「６」のスレッドを終了させる。

【００１８】

クラスライブラリが呼び出された際、クラスライブラリがどのアプリケーションから呼び出されているかをアプリケーション管理部１０３ｂに問い合わせる。クラスライブラリも、アプリケーションを実行しているスレッドまたは、そのスレッドから派生して生成されたスレッドで実行されているので、アプリケーション管理部１０３ｂは問い合わせ自体が実行されているスレッドからアプリケーションを特定しクラスライブラリに通知する。

【００１９】

例えば、アプリケーション管理部１０３ｂが図２（３）の状態で、クラスライブラリから呼び出されたとき、アプリケーション管理部１０３ｂは、その呼び出しを実行しているスレッドのスレッドＩＤを取得する。ここでスレッドＩＤは、オブジェクトのハッシュコードｗｏ用いるとすると、呼び出しで使用されたスレ

ッドのスレッドクラスのインスタンスを取得し、ハッシュコードを返す関数ｈａｓｈｃｏｄｅによって取得することができる。次に取得したスレッドＩＤを図２（３）のテーブルデータの中から検索し、アプリケーションＩＤ「２」に対応付けられて保持していることが判明するので、このアプリケーションＩＤ「２」を返す。

【００２０】

クラスライブラリ格納部１０３ｃは、システムが備えるデバイスを制御するためのクラスライブラリを格納し、具体的にはＲＯＭやハードディスクなどで構成される。クラスライブラリ格納部１０３ｃは、図中の「ライブラリ１」「ライブラリ２」、、のように、一般に複数のクラスライブラリを保持する。各クラスライブラリはＯＳ１０４のライブラリ格納部１０４ｂが格納するライブラリを呼び出し、システムが備えるデバイスを制御する。

【００２１】

クラスライブラリは、アプリケーションからリスナーと呼ばれるコールバック関数を登録され、これを特定のタイミングで呼び出す。クラスライブラリーは、リスナー登録される際、リスナー登録するアプリケーションに対して専用のスレッドを作成する。リスナー登録はＪａｖａアプリケーションのスレッドで実行されているので，このスレッド上で専用スレッドを作成すれば、自動的にアプリケーションのタスクのスレッドに組み込まれる。このスレッドはアプリケーション管理部１０３ｂでアプリケーションに対応付けられ管理される。リスナー呼び出し時は、この予め作成しておいたスレッドでリスナーを呼び出す。予め作成したスレッドを用いてリスナーを呼び出す具体的な方法の詳細は後述する。

【００２２】

ここで、１つのアプリケーションが複数回リスナー登録をする可能性がある。多くのスレッドを作りすぎないために、クラスライブラリはアプリケーション管理部１０３ｂに問い合わせることで、アプリケーションを特定し、アプリケーションに対して１つで専用スレッドだけを作成することが出来る。これを実現するためにクラスライブラリはアプリケーションと専用スレッドと登録されたリスナーを対応付けて保持する。

【００２３】

図３は、クラスライブラリが保持するアプリケーションと専用スレッドと登録されたリスナーの組の１例である。（１）では、既にアプリケーションＩＤ「１」のアプリケーションからリスナー「Ｌ１」と「Ｌ４」が登録され、これらのリスナーを呼び出すために、専用スレッド「ｔｈｒｅａｄ１０」がアプリケーションＩＤ「1」のアプリケーションから派生したスレッドとして生成されていることがわかる。ここで、新規にリスナー「Ｌ６」が登録されたとすると、クラスライブラリはアプリケーション管理部１０３ｂからアプリケーションのアプリケーションＩＤを取得する。ここで「２」が得られたとする。既に保持しているアプリケーションＩＤと比較し一致すれば、登録されるリスナーを登録するが、図３（１）を参照してい一致するものはないので、新規に専用スレッドを作成しアプリケーションＩＤと登録されたリスナーとともに保持する。図３（２）はこの状態を表す。

【００２４】

クラスライブラリ格納部１０３ｃが格納するクラスライブラリはアプリケーションの終了を通知される。この時、終了するアプリケーションのリスナー登録情報を削除する。

【００２５】

例えば図３（２）の状態で、アプリケーションＩＤ「１」のアプリケーションの終了が通知されると、このアプリケーションＩＤ「１」の情報を削除し図３（３）のようになる。

【００２６】

次に、予め作成したスレッドを用いてリスナーを呼び出す具体的な方法を図４を参照しながら説明する。一般にクラスライブラリに登録されたリスナーはシステムの状態が変化したときなどに呼び出される。ユーザーからのキー入力時に呼び出し等は典型的な例である。リスナーを呼び出すタイミングは一般にシステムの変化を知っているＯＳ１０４のライブラリ格納部１０４ｂに格納されるクラスライブラリから通知される。この通知は、ＯＳ１０４で生成されたタスク・スレッドで行われる。今、クラスライブラリが図３（２）の状態で２つの専用スレッ

ド生成しているとする。図４の中で、４０１は、専用スレッド「ｔｈｒｅａｄ１０」を、４０２は専用スレッド「ｔｈｒｅａｄ２０」を、４０３はクラスライブラリに通知を行うＯＳ１０４で生成されたスレッドを表している。アプリケーション用に生成された各専用スレッドに対してキュー４１１、４１２が用意される。クラスライブラリは、ＯＳ１０４のライブラリから通知された情報をキュー４１１、４１２に代入する。破線４２１、４２２は代入することを表し、４３１，４３２は代入された情報を表す。各専用スレッド４０１、４０２は、キュー４１１、４１２を監視し、キューの情報を発見すると、その情報を取り出し対応するリスナーを呼び出す。破線４４１，４４２はキューに代入された情報の取り出しを表す。図３（２）を参照して、専用スレッド４０１は、リスナー「Ｌ１」「Ｌ２」を呼び出し、専用スレッド４０２は、リスナー「Ｌ６」を呼び出す。

【００２７】

ＯＳ１０４は、Ｊａｖａミドルウエアを動作させることにより、Ｊａｖａアプリケーションを間接的に動作させ、カーネル１０４ａとライブラリ格納部１０４ｂが格納するライブラリで構成される。

【００２８】

カーネル１０４ａは、Ｊａｖａミドルウエア１０３のアプリケーション管理部１０３ｂから依頼され、Ｊａｖａアプリケーションを実行するタスクとスレッドを生成する。

【００２９】

カーネル１０４ａは、複数のスレッドを有するタスク単位で資源を管理する。スレッド上で実行されるアプリケーションから生成されるスレッドは、同じタスクのスレッドとして管理し、各スレッド上で実行されるアプリケーションによって確保される資源は、スレッドを含むタスクと対応付けて管理される。実際の資源の管理は、カーネル１０４ａが行う場合と，資源を直接供給するライブラリ格納部１０４ｂが格納するライブラリが行う場合の２通りがある。

【００３０】

また、カーネル１０４ａは、Ｊａｖａミドルウエア１０３のアプリケーション管理部１０３ｂからＪａｖａアプリケーションを終了するため、Ｊａｖａアプリ

ケーションに対応するタスクの通知を受けると、タスクに対応付けられて管理されている資源を全て回収する。

【００３１】

ライブラリ格納部１０４ｂは、システムが備えるデバイスを制御するためのライブラリを格納し、具体的にはＲＯＭやハードディスクなどで構成される。ライブラリ格納部１０４ｂは、図中の「ライブラリ１」「ライブラリ２」、、のように、一般に複数のライブラリを保持する。各ライブラリはハードウエア１０５のデバイスを制御する。

【００３２】

また、ライブラリは資源を供給すると共に、使用されている資源をタスク単位で管理する場合もある。この場合、資源供給の際にカーネル１０４ａｎｉ問い合わせ，資源を供給するタスクを問い合わせタスクと資源を組にして管理を行う。カーネル１０４ａより特定のタスクに関する資源回収の指示が有ると、指示されたタスクに対応する資源を回収する。

【００３３】

ハードウエア１０５は、ＯＳ１０４やＪａｖａミドルウエア１０３を動作させるＣＰＵ１０５ａ及びデバイス１０５ｂで構成される。

【００３４】

ＣＰＵ１０５は、ソフトウエアをカーネル１０４ａ、ライブラリ格納部１０４ｂが格納するライブラリ、ＶＭ部１０３ａ、クラスライブラリ格納部１０３ｂが格納するクラスライブラリ等のプログラムコードを実行する。

【００３５】

デバイス１０５ｂは、具体的には、メモリーデバイス、ディスプレイデバイス、入力デバイス（キーボード、リモコン等）、ファイルシステム、ネットワークデバイスなどが含まれ、それぞれのデバイスはクラスライブラリ格納部１０３ｃが格納するクラスライブラリ及びライブラリ格納部１０４ｂが格納するライブラリと通して、Ｊａｖａアプリケーションから制御可能である。

【００３６】

制御部１０６は、システム上で動作するアプリケーションの開始、終了などを

指示する。

【００３７】

アプリケーションの開始は、アプリケーション入力部１０１に、アプリケーションを受け付け、アプリケーション記憶部１０２に記憶する指示を行い、その後、アプリケーション管理部１０３ｂにアプリケーションを実行する指示を出す。

【００３８】

アプリケーションの終了は、アプリケーション管理部１０３ｂに実行しているアプリケーションの終了を指示する。アプリケーション管理部１０３ｂは、ＶＭ部１０３ａにアプリケーションの実行の停止を指示する。また、ＯＳ１０４のカーネル１０４ａにアプリケーションに対応するタスクを終了し、このタスクを構成するスレッドで確保された資源を全て回収する指示を与える。

【００３９】

図５は、本実施の形態のアプリケーション実行装置において、Ｊａｖａアプリケーションがクラスライブラリにリスナー登録する際のクラスライブラリの動作を説明するフローチャートである。

【００４０】

クラスライブラリは、リスナー登録のためのメソッドを呼ばれると、まずアプリケーション管理部１０３ｂからアプリケーションを特定するアプリケーションＩＤを取得する（ステップＳ５０１）。次に、リスナー、専用スレッド、アプリケーションＩＤを組にした登録情報の１つを取り出し（ステップＳ５０２）、登録情報のアプリケーションＩＤとアプリケーション管理部１０３ｂから取得したアプリケーションＩＤを比較する（ステップＳ５０３）。一致すれば、登録情報に登録されたリスナーを追加登録する（ステップＳ５０４）。不一致の場合は、全登録情報に関して比較を行い（ステップＳ５０５）、一致する登録情報がなければ、アプリケーション専用スレッドを生成する（ステップＳ５０６）。その後、リスナー、生成した専用スレッド、アプリケーションＩＤを組にした登録情報を作成し保持する（ステップＳ５０７）。

【００４１】

図６と図７は、クラスライブラリがＯＳ部１０４のライブラリからリスナー呼

び出しのタイミングを通知された際の、クラスライブラリのリスナー呼び出しの動作を説明するフローチャートである。

【００４２】

図６は、ＯＳ部１０４で生成されたライブラリからのスレッドで実行されるクラスライブラリの動作を表すフローチャートである。クラスライブラリは、ＯＳ部１０４のライブラリからリスナー呼び出しのタイミングを通知されると、リスナー、専用スレッド、アプリケーションＩＤを組にした登録情報の１つを取り出す（ステップＳ６０１）。登録情報の専用スレッドのキューにリスナー呼び出しに必要な呼出情報を代入する（ステップＳ６０２）。これを全登録情報に対して行う（ステップＳ６０３）。

【００４３】

図７は、専用スレッドで実行されるクラスライブラリの動作を表すフローチャートである。専用スレッドでは、キューに呼出情報が代入されていないかを監視している（ステップＳ７０１）。キューに呼出情報があれば、それを取り出す（ステップＳ７０２）。専用スレッドに対応するリスナーの１つを取り出し（ステップＳ７０３）、呼出情報に応じて登録されているリスナーを呼び出す（ステップＳ７０４）。これを全リスナーに対して行う（ステップＳ７０５）。

【００４４】

図８は、アプリケーション終了時のアプリケーション管理部１０３ｂの動作を説明するフローチャートである。

【００４５】

制御部１０６よりアプリケーション終了の指示を受けるとアプリケーション管理部１０３ｂは、カーネル１０４ａにタスクＩＤを指定し資源回収を依頼する（ステップＳ８０１）。カーネル１０４ａはＯＳ１０４内のライブラリ格納部１０４ｂ内のライブラリを呼び出すなどして、資源回収を行う。次に、アプリケーション管理部１０３ｂはクラスライブラリ格納部１０３ｃ内の各クラスライブラリにアプリケーションＩＤを指定しアプリケーションの終了を通知する（ステップＳ８０２）。各クラスライブラリは、指定されたアプリケーションＩＤを含む、リスナー、専用スレッド、アプリケーションＩＤを組にした登録情報を削除する

。最後に、アプリケーション管理部１０３ｂはアプリケーションを実行しているタスクを終了させる。

【００４６】

なお、ステップＳ８０１とステップＳ８０２は順序が逆でも実施可能である。

【００４７】

実施の形態１において、ＶＭ部１０３ａは、アプリケーションを実行するスレッド上で実行され、その中でＶＭ部１０３ａが動作するのに必要な資源も確保される。しかし、実施の形態１においては、これらの資源はアプリケーションの終了と共に回収されてしまうため、再度確保しなければならない。よって、システム起動時にＶＭ部１０３ａの専用スレッドを起動し、このスレッド上でＶＭ部１０３ａが必要な資源を確保することで、アプリケーション終了による資源再取得を不要にすることができる。

【００４８】

実施の形態１において、クラスライブラリ格納部１０３ｂが格納するクラスライブラリは、生成したＪａｖａアプリケーション用の専用スレッドでキューに呼び出し情報があるかどうかを監視している。しかし、監視を続けるとシステム全体の実行速度を下げる事になる。そこで、専用スレッドはウエイとしておき、ＯＳ１０４のライブラリからリスナー呼び出しのタイミングを通知されたとき、ウエイとしている専用スレッドを再起動してもよい。図９、図１０は、この場合のクラスライブラリの動作を表すフローチャートである。

【００４９】

図９は、ＯＳ部１０４で生成されたライブラリからのスレッドで実行されるクラスライブラリの動作を表すフローチャートである。クラスライブラリは、ＯＳ部１０４のライブラリからリスナー呼び出しのタイミングを通知されると、リスナー、専用スレッド、アプリケーションＩＤを組にした登録情報の１つを取り出す（ステップＳ６０１）。登録情報の専用スレッドのキューにリスナー呼び出しに必要な呼出情報を代入する（ステップＳ６０２）。その後、ウエイトしている専用スレッドを起動する（ステップＳ９０１）。これを全登録情報に対して行う（ステップＳ６０３）。

【００５０】

図１０は、専用スレッドで実行されるクラスライブラリの動作を表すフローチャートである。専用スレッドは、起動されると即座にウエイト状態になる（ステップＳ１００１）。ウエイト状態から起動されると、キューに呼出情報が代入されているはずなので、それを取り出す（ステップＳ７０２）。専用スレッドに対応するリスナーの１つを取り出し（ステップＳ７０３）、呼出情報に応じて登録されているリスナーを呼び出す（ステップＳ７０４）。これを全リスナーに対して行う（ステップＳ７０５）。リスナー呼び出しが終了すると再びウエイト状態に入る。

【００５１】

システムからＪａｖａアプリケーションのメソッドを呼び出すのはリスナーだけでなくＣｏｍｐｏｎｅｎｔクラスのｐｉａｎｔメソッドもある。このｐｉａｎｔメソッドもアプリケーションから派生するスレッドで呼び出すため、Ｃｏｍｐｏｎｅｎｔ生成時にｐａｉｎｔメソッドを呼び出す専用スレッドを生成する。

【００５２】

実施の形態１では、リスナー呼び出し時、また、実施の形態の変形例その３ではＣｏｍｐｏｎｅｎｔ生成時にアプリケーションのスレッドから派生した専用スレッドを作成した。しかし、アプリケーションがほぼ確実にリスナー登録するクラスライブラリには、アプリケーション起動時に専用スレッドを起動しても良い。これにより、クラスライブラリはリスナー登録時に毎回専用スレッドの有無を検査する必要が無くなり短時間でリスナー登録処理を完了することが出来る。

【００５３】

【発明の効果】

以上説明したように、本発明によれば、複数のスレッドで構成されるタスク単位で資源回収を行うＯＳと、Ｊａｖａアプリケーションを記憶するアプリケーション記憶手段と、前記アプリケーション記憶手段が記憶するＪａｖａアプリケーションを実行するＪａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅと、前記Ｊａｖａアプリケーションを前記ＯＳが生成するタスク及びスレッドと組にして管理し、終了するアプリケーションが使用している資源を前記ＯＳに回収する指示を出す

アプリケーション管理手段と、前記Ｊａｖａアプリケーション毎にスレッドを生成し、生成したスレッド上で前記アプリケーションのメソッドを呼び出すクラスライブラリを備えることとしている。

【００５４】

クラスライブラリは、アプリケーション毎にスレッドを生成することにより、Ｊａｖａミドルウエアでの資源回収を不要にし、ＯＳのみでのタスク・スレッドによる資源回収を可能にしている。これにより短時間で資源回収が可能になる。

【００５５】

更に、前記Ｊａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅは、タスク・スレッドを生成し、生成したスレッド上でＪａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅが使用する資源を確保することとしている。

【００５６】

これによりＪａｖａアプリケーション終了時にＪａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅが使用する資源が解放され、資源の再獲得をする処理を省くことが出来、アプリケーションの終了をより短かい時間で実行できる。

【００５７】

更に、前記アプリケーション管理手段は、スレッドによってＪａｖａアプリケーションを特定し、前記クラスライブラリは、前記Ｊａｖａアプリケーションに対して共通のスレッド生成し管理することとしている。

【００５８】

これにより、生成するスレッド数を減らすことが出来、システム全体のＪａｖａアプリケーション実行効率の低下を防ぐ。

【００５９】

更に、前記アプリケーション管理手段は、Ｊａｖａアプリケーションの終了を前記クラスライブラリに通知し、前記クラスライブラリは、アプリケーション終了時に保持しているＪａｖａアプリケーションに関するデータを削除することとしている。

【００６０】

クラスライブラリに余分な情報が残ることを防ぐことが出来る。

【００６１】

更に、前記ＯＳは、前記クラスライブラリがＪａｖａアプリケーションを呼び出すタイミングを通知し、前記クラスライブラリは、生成するスレッドに対応してキューを生成し、前記ＯＳからの通知によって、Ｊａｖａアプリケーションに通知する情報を前記キューに代入し、前記生成されたスレッドは前記キューに代入された情報を取り出し、Ｊａｖａアプリケーションを呼び出すこととしている。

【００６２】

キューを使用することで、必要な情報を伝達し、アプリケーション用に生成されたスレッド上でＪａｖａアプリケーションを呼び出すことが出来る。

【００６３】

更に、前記クラスライブラリは、Ｊａｖａアプリケーションからのリスナー登録時に前記Ｊａｖａアプリケーションに対応するスレッドを生成することとしている。

【００６４】

リスナー登録時にスレッドを生成することで、必要時のみスレッドを生成することが出来る。

【００６５】

更に、前記クラスライブラリは、ＪａｖａアプリケーションがＣｏｍｐｏｎｅｎｔクラスのインスタンスを生成する時に前記Ｊａｖａアプリケーションに対応するスレッドを生成することとしている。

【００６６】

Ｃｏｍｐｏｎｅｎｔのインスタンス生成時にスレッドを生成することで、必要時のみスレッドを生成することが出来る。

【００６７】

更に、前記クラスライブラリは、Ｊａｖａアプリケーションの起動時に前記Ｊａｖａアプリケーションに対応するスレッドを生成することとしている。

【００６８】

高い確率でリスナー登録されるクラスライブラリではアプリケーション起動時

にスレッドを作成しておくことで、リスナー登録の処理を短時間で実行できる。

【００６９】

ことを特徴とするＪａｖａアプリケーション実行装置。

【００７０】

更に、コンピュータ読み取り可能な記録媒体であって、複数のスレッドで構成されるタスク単位で資源回収を行うＯＳと、Ｊａｖａアプリケーションを記憶するアプリケーション記憶手段と、前記アプリケーション記憶手段が記憶するＪａｖａアプリケーションを実行するＪａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅと、前記Ｊａｖａアプリケーションを前記ＯＳが生成するタスク及びスレッドと組にして管理し、終了するアプリケーションが使用している資源を前記ＯＳに回収する指示を出すアプリケーション管理手段と、前記Ｊａｖａアプリケーション毎にスレッドを生成し、生成したスレッド上で前記アプリケーションのメソッドを呼び出すクラスライブラリの各機能を発揮するプログラムとすることとしている。このような構成によって、資源回収機構を有するアプリケーション実行装置としてコンピュータを利用することができる。

【図面の簡単な説明】

【図１】

本発明に係るＯＳによる資源回収機能を有するＪａｖａアプリケーション実行装置の実施の形態１の構成図

【図２】

（１）実施の形態１に含まれるアプリケーション管理部１０３ｂが、アプリケーションＩＤ、タスクＩＤ、スレッドＩＤの組を管理している状態を表す図

（２）実施の形態１に含まれるアプリケーション管理部１０３ｂが、アプリケーションＩＤ、タスクＩＤ、スレッドＩＤの組を管理している状態を表す図

（３）実施の形態１に含まれるアプリケーション管理部１０３ｂが、アプリケーションＩＤ、タスクＩＤ、スレッドＩＤの組を管理している状態を表す図

【図３】

（１）実施の形態１において、クラスライブラリがアプリケーションＩＤ、スレッドインスタンス、リスナーインスタンスの組を管理している状態を表す図

（２）実施の形態１において、クラスライブラリがアプリケーションＩＤ、スレッドインスタンス、リスナーインスタンスの組を管理している状態を表す図

（３）実施の形態１において、クラスライブラリがアプリケーションＩＤ、スレッドインスタンス、リスナーインスタンスの組を管理している状態を表す図

【図４】

実施の形態１において、クラスライブラリが登録されたリスナーのメソッドを呼び出す手順とＯＳのスレッド、アプリケーション用に生成したスレッド及びそのキューの関係を示す図

【図５】

実施の形態１において、クラスライブラリはアプリケーション専用のスレッドを生成し管理する際の動作を表すフローチャート

【図６】

実施の形態１において、クラスライブラリがリスナーを呼び出す際、ＯＳのスレッドで実行する動作を表すフローチャート

【図７】

実施の形態１において、クラスライブラリがリスナーを呼び出す際、アプリケーション専用に生成したスレッドで実行する動作を表すフローチャート

【図８】

実施の形態１において、アプリケーション管理部１０３ｂがアプリケーション終了時に行う資源回収の動作を表すフローチャート

【図９】

上記実施の形態１において、クラスライブラリがリスナーを呼び出す際、ＯＳのスレッドで実行する動作を表すフローチャート

【図１０】

上記実施の形態１において、クラスライブラリがリスナーを呼び出す際、アプリケーション専用に生成したスレッドで実行する動作を表すフローチャート

【図１１】

従来のアプリケーション実行装置が管理する資源の管理方法を示す図

【符号の説明】

１０１　アプリケーション入力部

１０２　アプリケーション記憶部

１０３　Ｊａｖａミドルウエア

１０３ａ　ＶＭ部

１０３ｂ　アプリケーション管理部

１０３ｃ　クラスライブラリ格納部

１０４　ＯＳ

１０４ａ　カーネル

１０４ｂ　ライブラリ格納部

１０５　ハードウエア

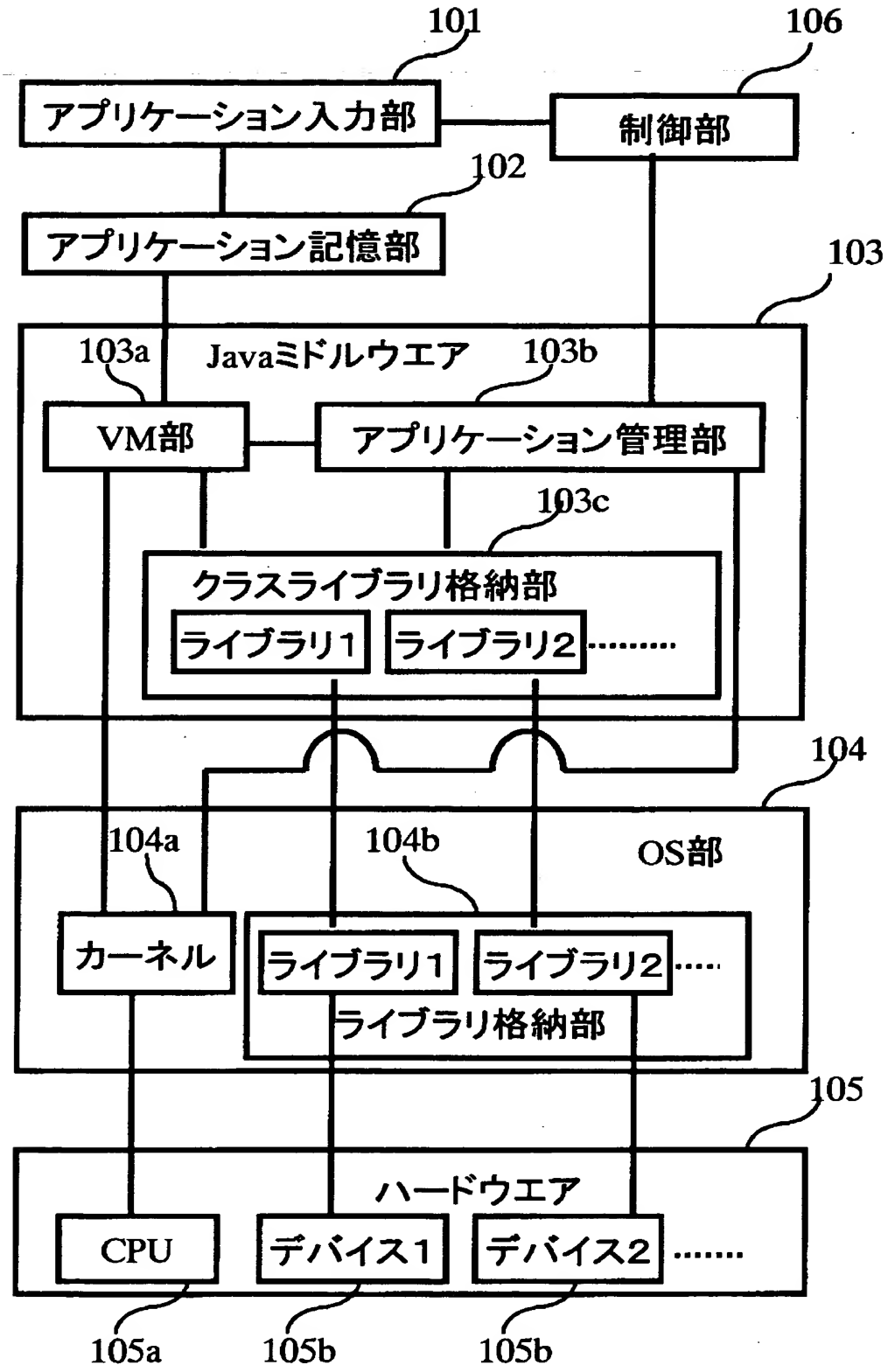
１０５ａ　ＣＰＵ

１０５ｂ　デバイス

１０６　制御部

【書類名】　　図面

【図１】


101
106
アプリケーション入力部
制御部
102
アプリケーション記憶部
103
103a
Javaミドルウエア
103b
VM部
アプリケーション管理部
103c
クラスライブラリ格納部
ライブラリ1
ライブラリ2
104
104a
104b
OS部
カーネル
ライブラリ1
ライブラリ2
ライブラリ格納部
105
ハードウエア
CPU
デバイス1
デバイス2
105a
105b
105b

【図２】

(1)

アプリケーションID	タスクID	Thread ID
1	201	1,2
2	202	4,5,6

(2)

アプリケーションID	タスクID	Thread ID
1	201	1,2,7
2	202	4,5,6

(3)

アプリケーションID	タスクID	Thread ID
1	201	1,2,7
2	202	4,6

【図３】

(1)

アプリケーションID	スレッドインスタンス	リスナーインスタンス
1	thread10	L1,L4

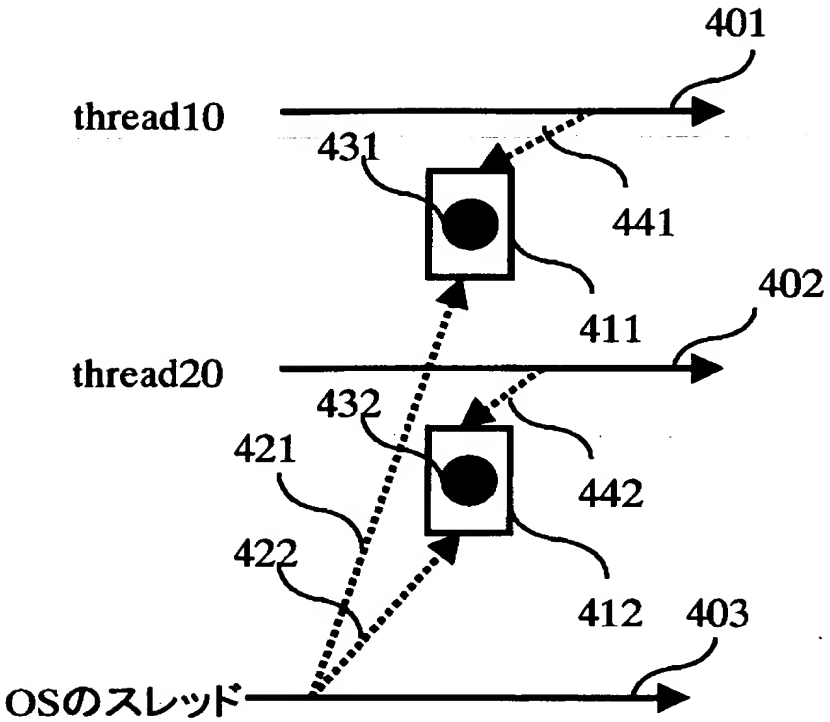
(2)

アプリケーションID	スレッドインスタンス	リスナーインスタンス
1	thread10	L1,L4
2	thread20	L6

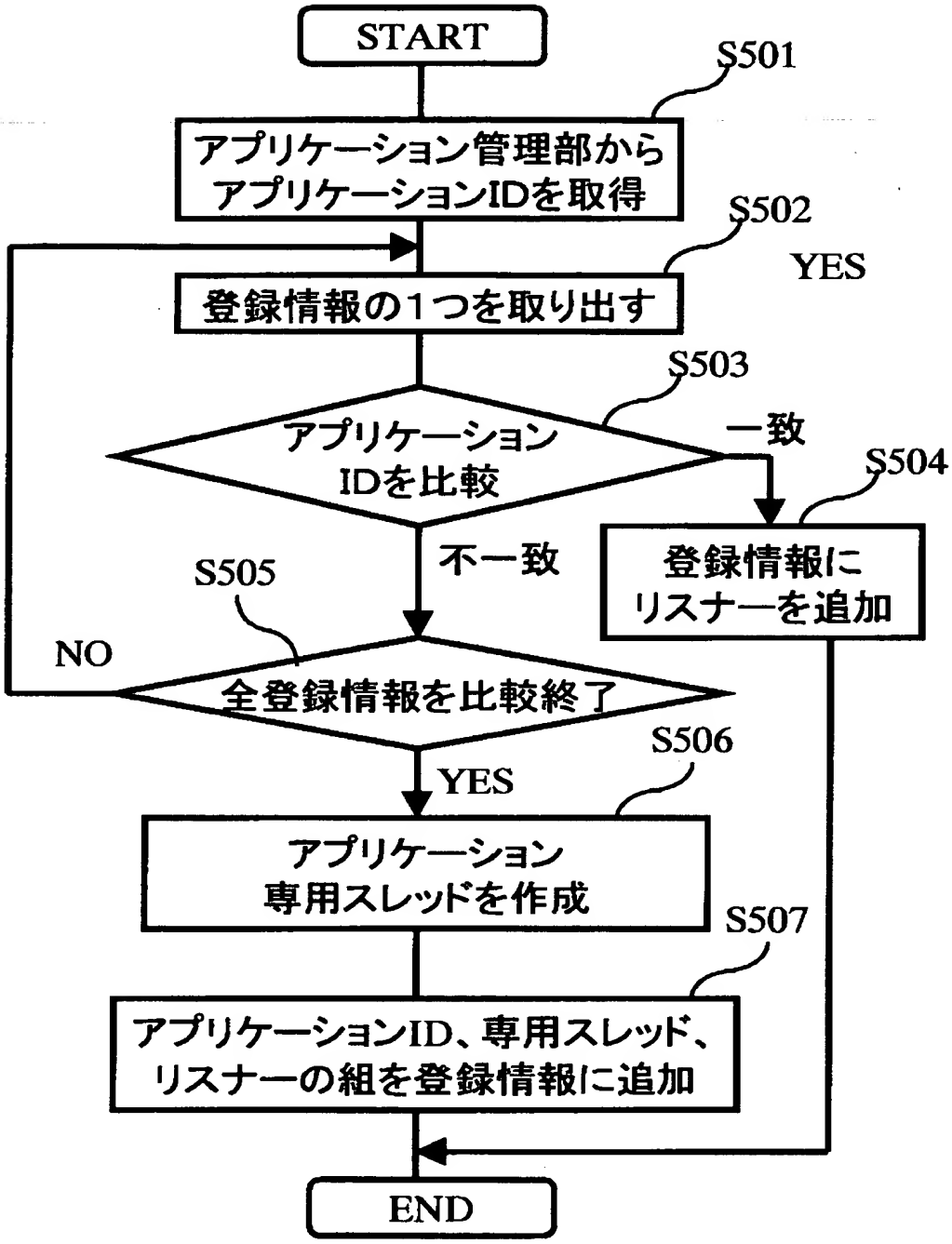
(3)

アプリケーションID	スレッドインスタンス	リスナーインスタンス
2	thread20	L6

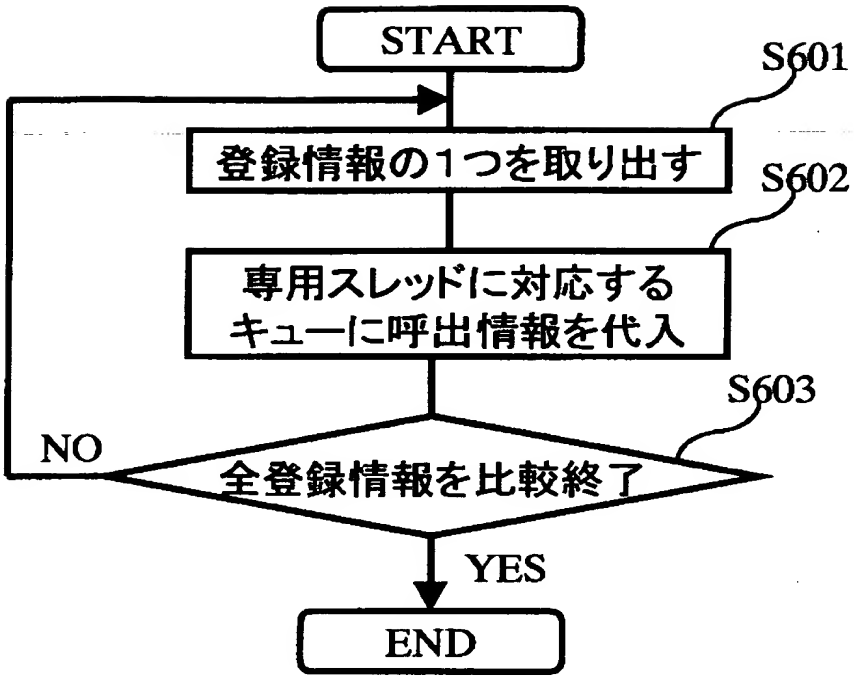
【図４】


401
thread10
431
441
402
411
thread20
432
421
442
422
412
403
OSのスレッド

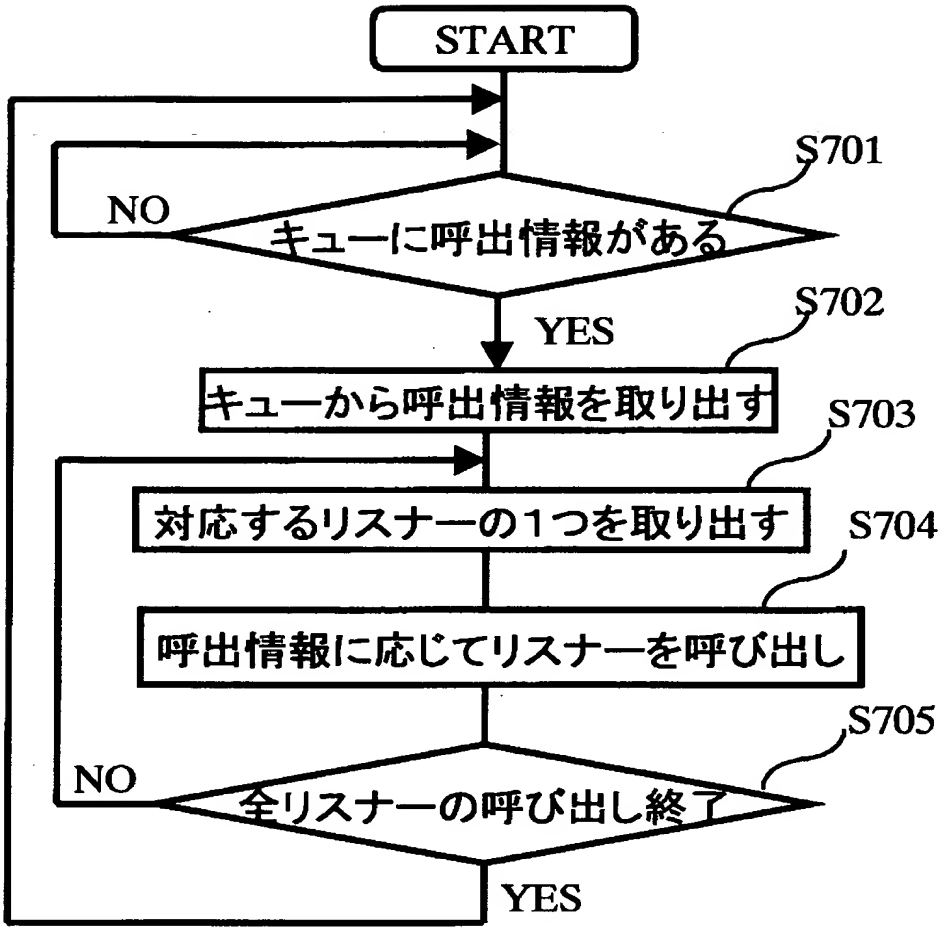
【図５】


START
S501
アプリケーション管理部から
アプリケーションIDを取得
S502
YES
登録情報の１つを取り出す
S503
アプリケーション
IDを比較
一致
S504
登録情報に
リスナーを追加
不一致
S505
NO
全登録情報を比較終了
S506
YES
アプリケーション
専用スレッドを作成
S507
アプリケーションID、専用スレッド、
リスナーの組を登録情報に追加
END

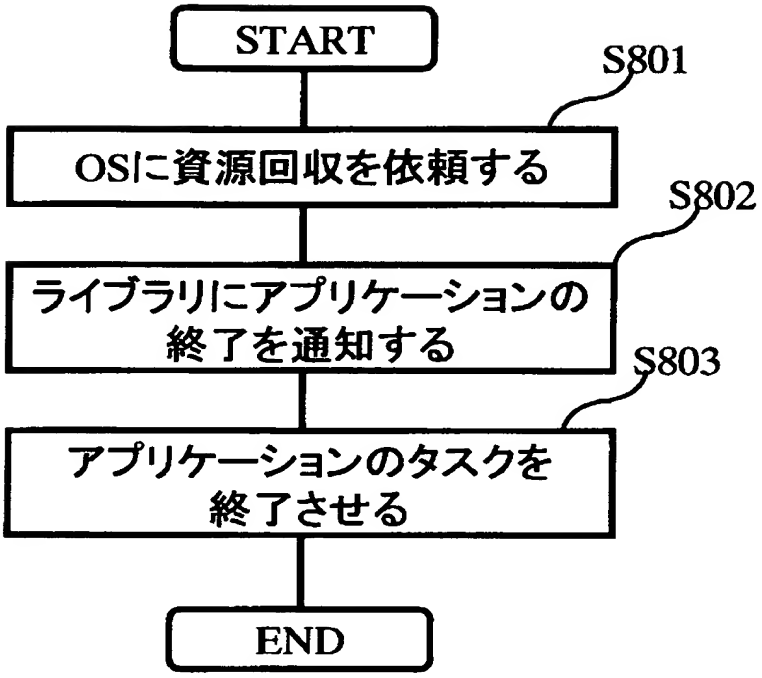
【図６】


START
S601
登録情報の１つを取り出す
S602
専用スレッドに対応する
キューに呼出情報を代入
S603
NO
全登録情報を比較終了
YES
END

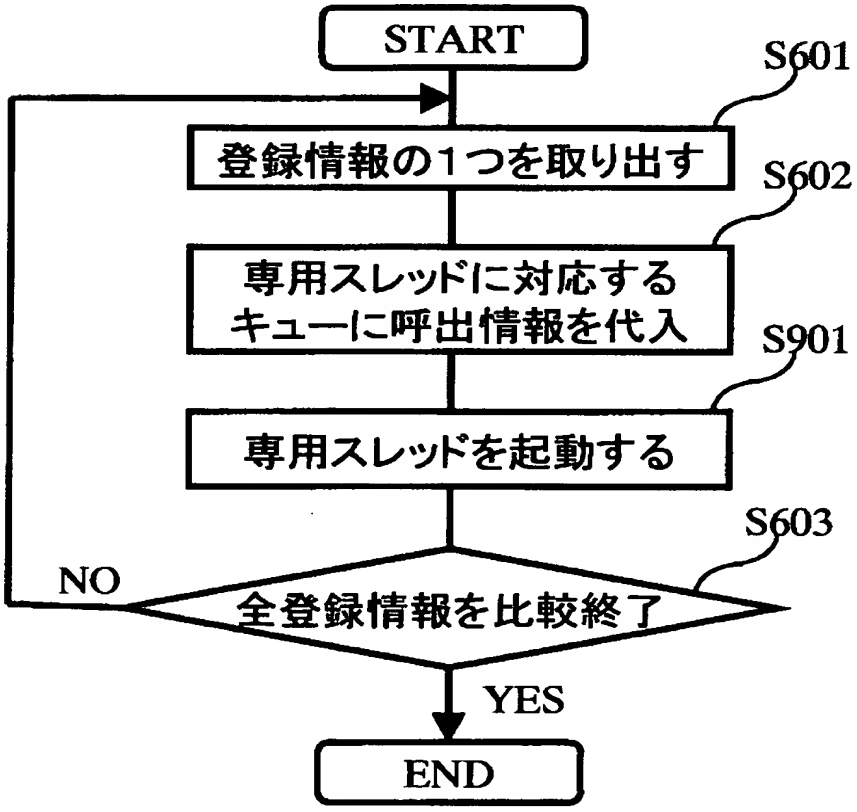
【図７】


START
S701
NO
キューに呼出情報がある
YES
S702
キューから呼出情報を取り出す
S703
対応するリスナーの１つを取り出す
S704
呼出情報に応じてリスナーを呼び出し
S705
NO
全リスナーの呼び出し終了
YES

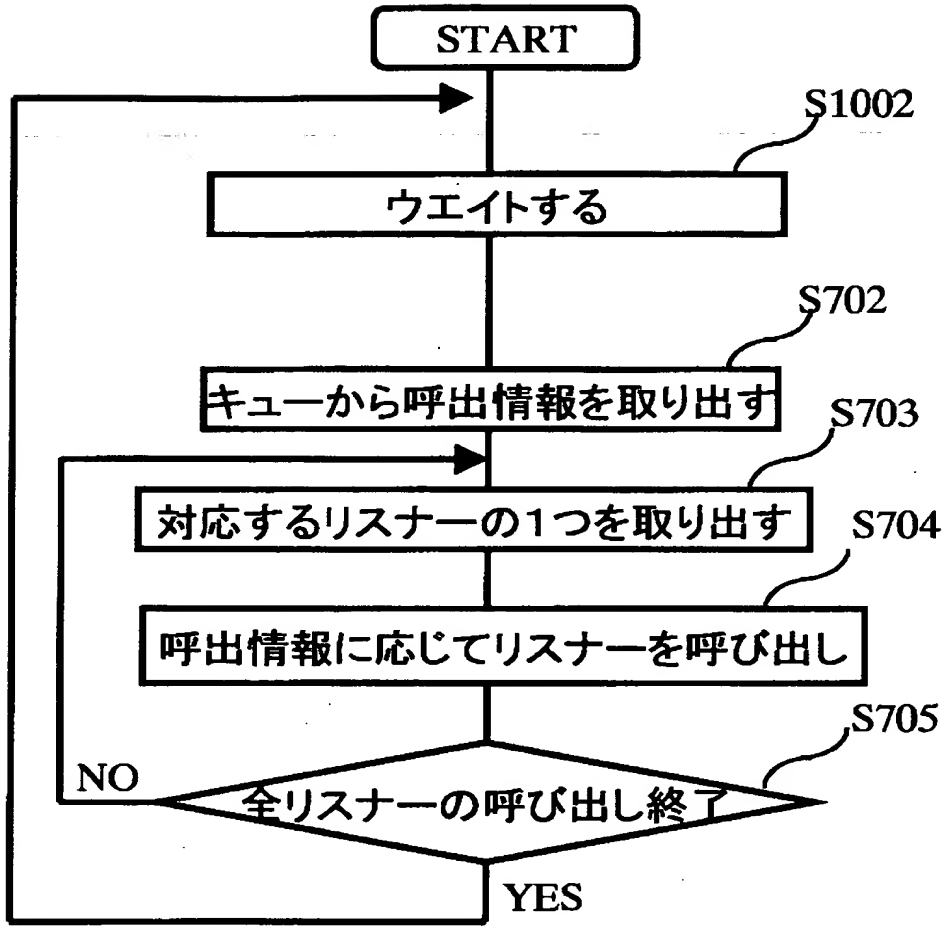
【図８】


START
S801
OSに資源回収を依頼する
S802
ライブラリにアプリケーションの終了を通知する
S803
アプリケーションのタスクを終了させる
END

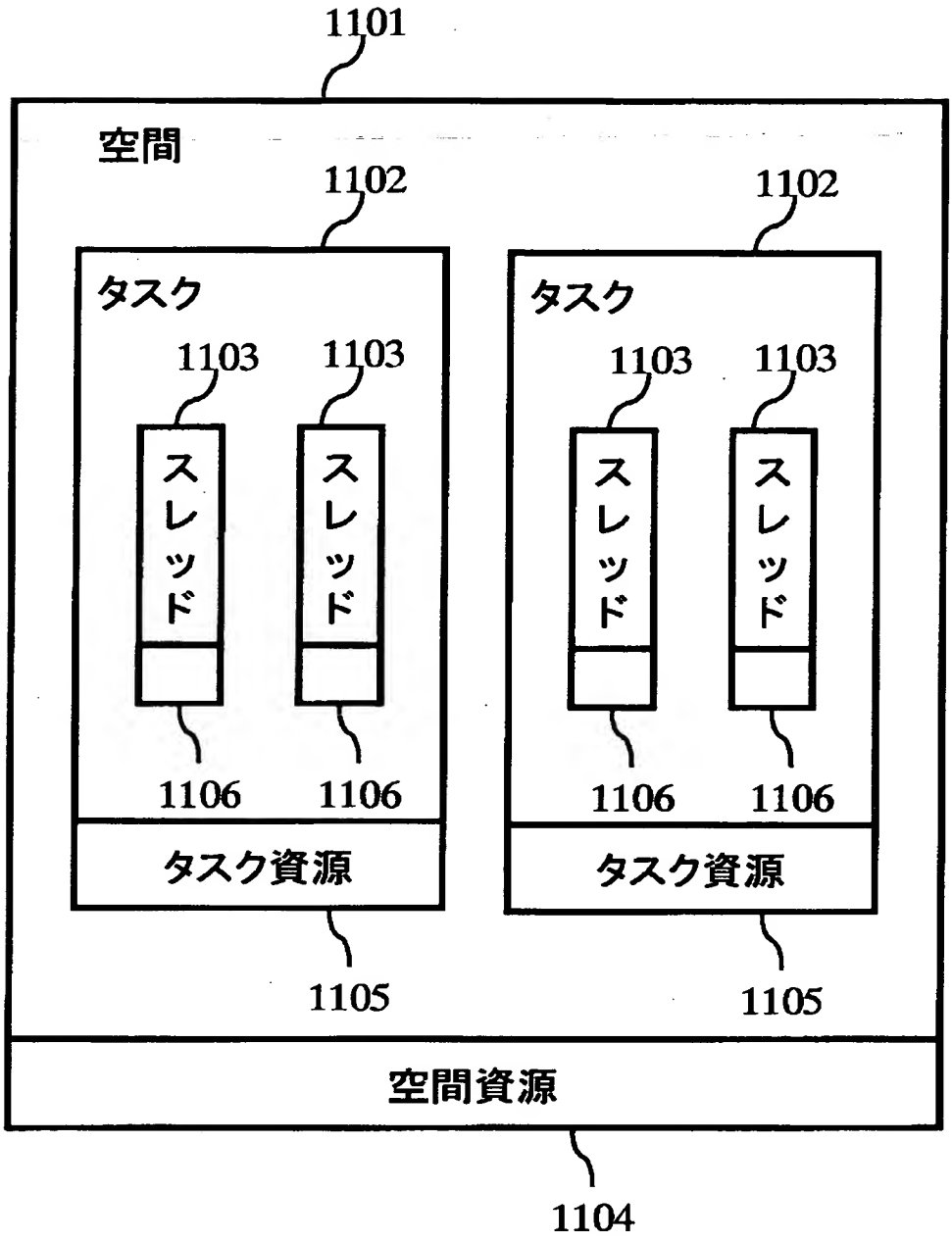
【図９】


START
S601
登録情報の１つを取り出す
S602
専用スレッドに対応するキューに呼出情報を代入
S901
専用スレッドを起動する
S603
NO
全登録情報を比較終了
YES
END

【図１０】


START
S1002
ウエイトする
S702
キューから呼出情報を取り出す
S703
対応するリスナーの１つを取り出す
S704
呼出情報に応じてリスナーを呼び出し
S705
NO
全リスナーの呼び出し終了
YES

【図１１】


1101
空間
1102
1102
タスク
タスク
1103
1103
1103
1103
スレッド
スレッド
スレッド
スレッド
1106
1106
1106
1106
タスク資源
タスク資源
1105
1105
空間資源
1104

【書類名】　要約書

【要約】

【課題】　Ｊａｖａミドルウエアを再起動せずに次のアプリケーションを実行すると資源回収が不十分なため、資源不足でアプリケーションが動作しなくなる。

【解決手段】　Ｊａｖａアプリケーション管理部１０３ｂはタスクとスレッドをＯＳ１０４のカーネル１０４ａに依頼して用意し、そのスレッド上でＶＭ部１０３ａはＪａｖａアプリケーションを実行する。Ｊａｖａミドルウエア１０３のクラスライブラリーはＪａｖａアプリケーションからリスナー登録される際、Ｊａｖａアプリケーションに対応するスレッドを生成する。ＯＳ１０４のライブラリからリスナーを呼び出すタイミングを通知された時、クラスライブラリは、生成しておいたスレッド上でリスナーを呼び出す。アプリケーション終了時、Ｊａｖａアプリケーション管理部１０３ｂはカーネル１０４ａに依頼してＪａｖａアプリケーションが使用している資源を回収する。

【選択図】　図１

出　願　人　履　歴　情　報

識別番号　　〔０００００５８２１〕

１．変更年月日　　１９９０年　８月２８日
［変更理由］　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社